Software-Projekt I

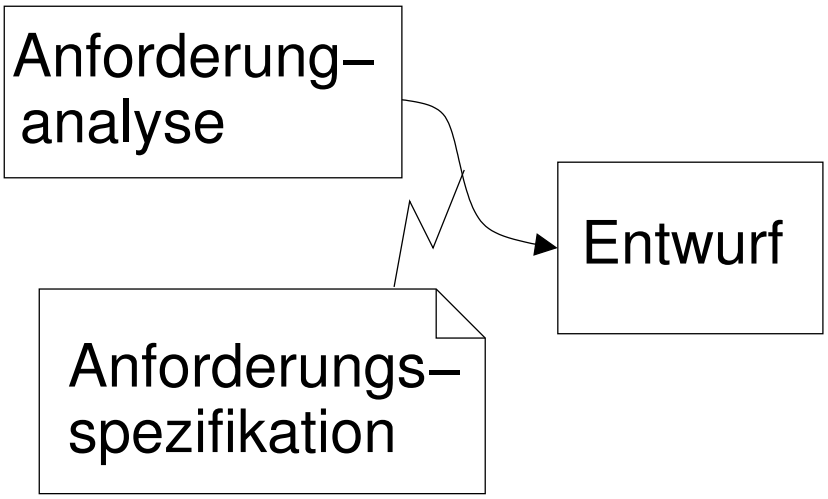
Prof. Dr. Rainer Koschke

Arbeitsgruppe Softwaretechnik
Fachbereich Mathematik und Informatik
Universität Bremen

Sommersemester 2013

Software-Prüfung mit Reviews I

Motivation



- Entwicklung = Sequenz von Aktivitäten mit Zwischenprodukten (Anforderungsspezifikation, Entwurf, Code, Testfälle etc.)
- Qualität der Zwischenprodukte bestimmt wesentlich Qualität der von ihnen abhängigen Aktivitäten

→ die Qualität der Zwischenprodukte muss gesichert werden

→ wir müssen sicher stellen, dass alle Beteiligten sich über das Zwischenprodukt einig sind

- Quellcode kann durch Tests überprüft werden
- aber was machen wir mit nicht direkt ausführbaren Dokumenten wie Anforderungsspezifikation und Entwurf?

Fragen

- Welche Möglichkeiten gibt es, Software zu prüfen?
- Im Allgemeinen: Wie führt man Reviews beliebiger Dokumente durch?
- Im Speziellen: Wie führt man Reviews der Anforderungsspezifikation durch?

	. . . ist prüfbar durch	
das Dokument . . .	Inspektion	Test
Lastenheft	X	
Pflichtenheft	X	
Systementwurf	X	
Definition der Daten und Algorithmen	X	
Benutzerhandbuch	X	
Testdaten	X	
Code	X	X
Anleitungen etc.	X	

Naturgemäß lassen sich mechanische Prüfungen nur dann anwenden, wenn die zu prüfenden Informationen einer mechanischen Analyse zugänglich sind. Darum kommt in der Software-Entwicklung als Prüfverfahren ganz überwiegend die Inspektion in Frage.

Reviews (Frühauf u. a. 2000)

Prinzip

- Eine Software-Einheit wird (dezentral) von mehreren Gutachtern inspiziert;
- in einer gemeinsamen Sitzung werden die Mängel zusammengetragen und dokumentiert.

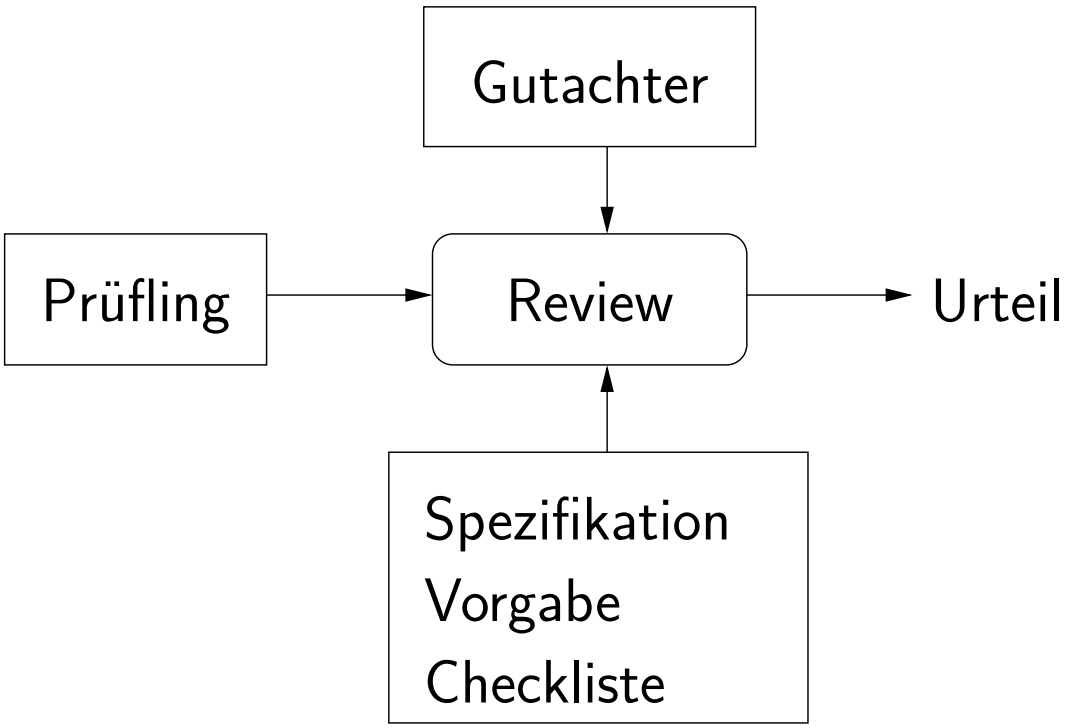
Ziel

- Fehler zu finden,
- **nicht**, die Fehler auch zu beheben.

Nachfolgend wird das technische Review beschrieben. In der Praxis gibt es zahlreiche Varianten. Zwischen diesen Formen gibt es keine simple Qualitätsordnung! Das heißt, wir wissen nicht, welche davon effektiver und effizienter ist in der Fehlersuche.

Fehlerbehebung ist ein separater Arbeitsschritt, den der Entwickler im Allgemeinen wieder ohne Mitwirkung Dritter durchführt.

Reviews (Frühauf u. a. 2000)



Prüfling

- kann jeder in sich abgeschlossene, für Menschen lesbare Teil von Software sein
- Beispiele: ein einzelnes Dokument, ein Codemodul, ein Testfall

Voraussetzung:

- Referenzunterlagen benötigt, die eine Beurteilung erlauben
- dazu gehören eine Vorgabe oder Spezifikation sowie die relevanten Richtlinien
- zusätzlich können Fragenkataloge (Checklisten) verwendet werden

Rollen im Review (Frühauf u. a. 2000)

Moderator

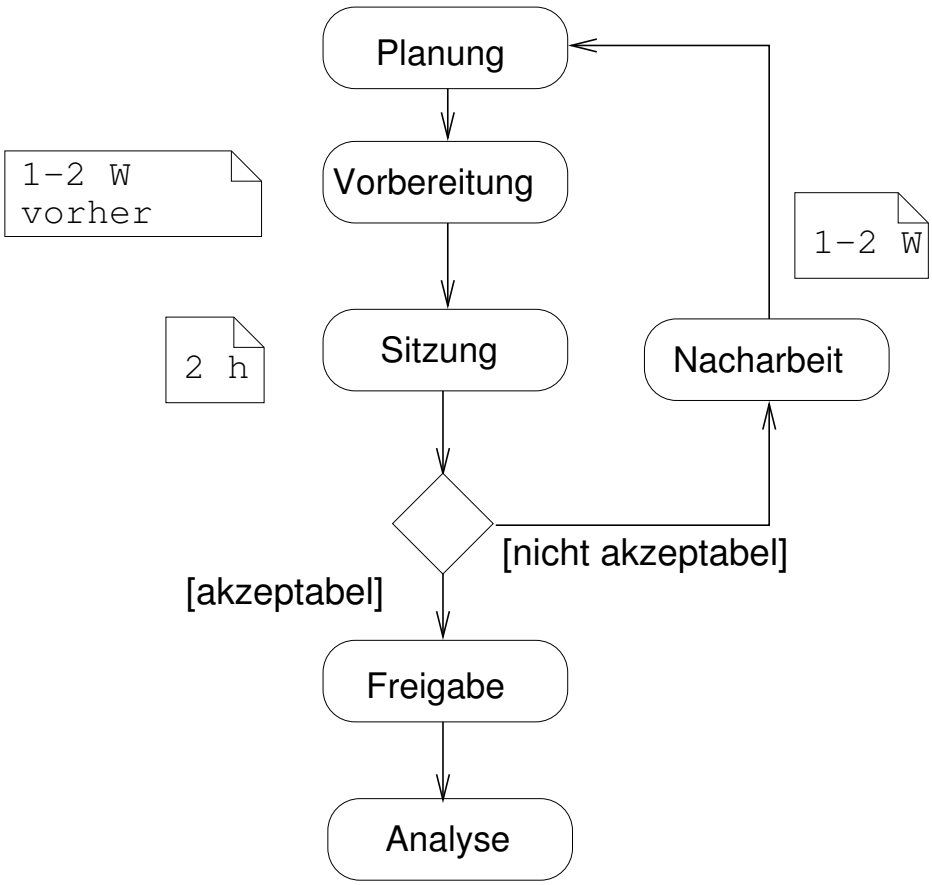
Autor

Sekretär

Gutachter

- **Moderator** leitet das Review, ist also für den ordnungsgemäßen Ablauf verantwortlich.
- **Sekretär** führt das Protokoll.
- **Gutachter** sind Experten, die den Prüfling beurteilen können, z.B. andere Entwickler, Benutzer, Auftraggeber, Produkt-Manager, Tester, Architekten, Handbuchautoren, Wartungspersonal.
- **Autor** ist der Urheber des Prüflings oder ein Repräsentant des Teams, das den Prüfling erstellt hat.

Review-Prozess



- Im ersten Schritt des Reviews, in der Vorbereitung, müssen die Gutachter das zu prüfende Dokument lesen und nach bestimmten, ihnen individuell zugeteilten Gesichtspunkten prüfen.
- Den zweiten Schritt bildet die Review-Sitzung, in der die Gutachter ihre in der Vorbereitung gefundenen Probleme vorbringen. Gemeinsam erheben, gewichten und protokollieren sie die Befunde. Der Auftrag für die Review-Sitzung lautet also, das Dokument oder Programm zu begutachten und nicht zu korrigieren, d.h. die Sitzung hat das Ziel, Probleme aufzuzeigen, nicht sie zu lösen.
- Die Nacharbeit selbst wird dem Autor oder den Autoren zugeteilt. Die Gutachter kommen bei der Überprüfung der Nacharbeit evtl. nochmals zum Zug.

Gutachter im Review

- andere Entwickler,
- Benutzer,
- Auftraggeber,
- Produkt-Manager,
- Tester,
- Architekten,
- Handbuchautoren,
- Wartungspersonal.

Beim technischen Review werden Kollegen als Gutachter beigezogen. Dafür kommen vor allem andere Entwickler in Frage, aber auch Benutzer, Produkt-Manager oder Auftraggeber. Die Gutachter erhalten im technischen Review konkrete Aufträge.

Aufgaben der Gutachter im Review

- Vorbereitung
- Sitzung
  - begutachten
  - **nicht** korrigieren
- Nacharbeit

- Vorbereitung: Prüfling lesen und nach bestimmten, ihnen individuell zugeteilten Gesichtspunkten prüfen.
- Sitzung: gefundene Probleme vorbringen; Befund gemeinsam erheben, gewichten und protokollieren
  - begutachten
  - **nicht** korrigieren
- Nacharbeit: Autor korrigiert nach Auflagen der Gutachter
- eventuell weitere Iteration mit Gutachtern

Review-Regeln

- Review-Sitzung ist auf 2h beschränkt.
- Moderator kann Sitzung absagen oder abbrechen.
- Moderator ≠ Gutachter.
- Jeder Gutachter bekommt Gelegenheit, seine Befunde angemessen zu präsentieren.
- Der Prüfling – nicht der Autor – steht zur Diskussion.

Falls nötig, wird eine weitere Sitzung, frühestens am nächsten Tag, einberufen. Grund des Abbruchs ist zu protokollieren.

- z.B. weil Gutachter nicht erscheinen oder ungenügend vorbereitet sind

Das heißt:

- Gutachter müssen auf ihre Ausdrucksweise achten.
- Autor darf weder sich noch das Resultat verteidigen.

Review-Regeln II

- Stilfragen (außerhalb der Richtlinien) werden nicht diskutiert.
- Review-Team identifiziert Schwachpunkte und Fehler, nicht aber Verbesserungen.
- Befunde werden gewichtet als:
  - kritischer Fehler
  - Hauptfehler
  - Nebenfehler
  - gut

- Team macht keine konstruktiven Vorschläge, wie Mängel zu beheben sind;
- Befunde werden nicht in der Form von Anweisungen an den Autor protokolliert.

Der Konsens der Gutachter zu einem Befund wird laufend protokolliert. Die einzelnen Befunde werden gewichtet als

- kritischer Fehler (Prüfling für den vorgesehenen Zweck unbrauchbar, Fehler muss vor der Freigabe behoben werden)
- Hauptfehler (Nutzbarkeit des Prüflings beeinträchtigt, Fehler sollte vor der Freigabe behoben werden),
- Nebenfehler (beeinträchtigen den Nutzen kaum),
- gut (fehlerfrei, bei Überarbeitung nicht ändern)

Review-Regeln III

- Review-Team gibt Empfehlung über die Annahme des Prüflings:
  - akzeptieren ohne Änderungen
  - akzeptieren mit Änderungen (kein weiteres Review)
  - nicht akzeptieren (weiteres Review erforderlich)
- Alle Sitzungsteilnehmer unterschreiben das Protokoll.

Checkliste für ein Anforderungsdokument

Aspekt Form: Ist die Darstellung im Dokument sinnvoll?

- a1 Sind Anforderungen als solche erkennbar, d.h. von Erklärungen unterscheidbar?
- a2 Sind alle Anforderungen eindeutig referenzierbar?
- a3 Ist die Spezifikation jeder Anforderung eindeutig?
- a4 Sind alle Anforderungen überprüfbar formuliert?

Checkliste für ein Anforderungsdokument II

Aspekt Vertraulichkeit: Sind die wesentlichen Aspekte des Datenschutzes berücksichtigt?

- d1 Ist spezifiziert, welche Information vertraulich zu behandeln ist?
- d2 Sind die Zugriffsrechte aller Benutzerklassen definiert?
- d3 Ist definiert, gegen welche Art von unberechtigtem Zugriff die Information geschützt werden muss?

Checklisten für ein Anforderungsdokument III

Aspekt Schnittstellen: Sind alle Schnittstellen eindeutig spezifiziert?

- b1 Sind alle Objekte der Umgebung (Benutzer, andere Systeme, Basis-Software etc.) sowie alle Informationsflüsse von und nach diesen Objekten spezifiziert?
- b2 Sind alle Benutzerklassen (Dauerbenutzer, gelegentliche Benutzer, System-Administrator, etc.) des Systems identifiziert?
- b3 ...
- b7 Sind Vorgaben gemacht bezüglich Verwendung von Betriebssystem-Funktionen, Bibliotheken und Hilfsprogrammen?

Varianten des Software-Reviews

Design and Code Inspection (nach Fagan)

Moderator

Autor

Sekretär

Gutachter

- mit Einführungssitzung,
- Gutachter-Notizen, die abgegeben werden,
- Vorleser,
- Entscheidungskompetenz,
- Metriken-Erhebung.

Varianten des Software-Reviews

Structured Walkthrough

Moderator

Autor

Sekretär

Gutachter

- ist die Billig-Variante des Reviews: Autor ist Moderator;
- er kompensiert durch seine Präsentation die Einsparung (oder Reduktion) der Vorbereitung;
- während seiner Vorbereitung entdeckt er selbst viele Fehler.

Varianten des Software-Reviews

Stellungnahme

Moderator

Autor

Sekretär

Gutachter

- ist ein „off-line“–Review unter der Regie des Autors;
- den Vorteilen (geringer Organisationsaufwand) stehen erhebliche Nachteile gegenüber (Prüfling nicht vor Weiterbearbeitung geschützt, Qualität der Prüfung und Umsetzung der Resultate nicht kontrolliert).

Varianten des Software-Reviews

Schreibtischtest (besser: die Selbstkontrolle)

Moderator

Autor

Sekretär

Gutachter

- führt jeder Entwickler allein durch (vgl. Humphreys Personal Software Process);
- ersetzt die eigentliche Prüfung nicht.

Varianten des Software-Reviews

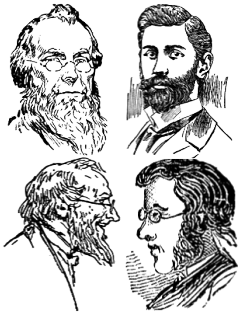
Perspektivisches Review

Moderator

Autor

Sekretär

Gutachter



- Prinzip:
  - Prüfling wird von Gutachtern mit unterschiedlichem Interesse geprüft
  - Gutachter haben konkreten Auftrag: Prüfling (hypothetisch) benutzen
- Beispiel:
  - Tester → Testfälle entwickeln
  - Benutzer → System benutzen
  - Handbuchautor → Handbuch schreiben
  - Architekt → Entwurf erstellen

Varianten des Software-Reviews

Eine Vielzahl empirischer Studien legt nahe:

- Reviews, die auf Checklisten basieren, sind effektiver als Ad-Hoc-Reviews;
- perspektivische Reviews sind effektiver und effizienter als Reviews, die auf Checklisten basieren:
  → mehr Fehler/Mängel in gleicher Zeiteinheit gefunden

(Berling und Runeson 2003; Fusaro u. a. 1997; Hayes 1999; Miller u. a. 1998; Porter u. a. 1995; Porter und Votta 1998; Sandahl u. a. 1998)

Woran scheitern Reviews? Fallen und Gegenmittel

Probleme bei der Ein-/Durchführung von Reviews:

- Gute Moderatoren fehlen
- Bezugsdokumente fehlen
- Entwickler haben Angst
- Gutachter beißen sich an Äußerlichkeiten fest
- Zeitdruck sabotiert Prüfungen
- Zu prüfende Dokumente wurden verändert
- Interesse an Reviews kühlt sich ab („Jetzt ist doch eigentlich alles in Ordnung!“)

→ Leute aussuchen und ausbilden

→ suchen, anpassen, bereitstellen

→ erste Reviews gründlich vorbereiten (und auf die Ängste eingehen, selbst wenn sie geleugnet werden)

→ Bedeutung der Äußerlichkeiten klarstellen, aber dann weitergehen;
→ beim zweiten Review sicherstellen, dass die Äußerlichkeiten in Ordnung sind.

→ klare Entscheidungen über die Prioritäten treffen

→ Konfigurationsmanagement verbessern

→ Statistiken führen, Erfolg nachweisen

Wiederholungsfragen

- Was ist der Unterschied zwischen Verifikation und Validierung?
- Welche Arten der Software-Prüfung gibt es?
- Was ist ein Review? Welche Rollen sieht es vor? Wie wird es durchgeführt?
- Welche Review-Varianten gibt es?
- Worauf ist bei Reviews zu achten?

Weiterführende Literatur

- Frühauf u. a. (2000) führen in Software-Prüfung ein.
- Shull u. a. (2000) zeigen in ihren empirischen Untersuchungen, dass perspektivische Reviews effektiver sein können als Reviews, die nur auf allgemeinen Check-Listen basieren

1 **Berling und Runeson 2003** Berling, T. ; Runeson, P.: Evaluation of a perspective based review method applied in an industrial setting. In: IEE Proceedings 150 (2003), Juni, Nr. 3

2 **Frühauf u. a. 2000** Frühauf ; Ludewig ; Sandmayr: Software-Prüfung. 4. Auflage. vdf Zürich, 2000

3 **Fusaro u. a. 1997** Fusaro, P. ; Lunibile, F. ; Visaggio, G.: A replicated experiment to assess requirements inspection techniques. In: Empirical Software Engineering 2 (1997), S. 39–57

4 **Hayes 1999** Hayes, W.: Research synthesis in software engineering: a case for meta-analysis. In: Proc. 6th Int. Symp. on Software Metrics, IEEE Computer Society Press, November 1999, S. 143–151

5 **Miller u. a. 1998** Miller, J. ; Wood, M. ; Roper, M.: Further experiences with scenarios and checklists. In: Empirical Software Engineering 3 (1998), S. 37–64

6 Porter und Votta 1998 Porter, A. ; Votta, L.: Comparing detection methods for software requirements inspection: a replication using professional subjects. In: Empirical Software Engineering 3 (1998), S. 355–379

7 Porter u. a. 1995 Porter, A. ; Votta, L. ; Basili, V.: Comparing detection methods for software requirements inspection: a replicated experiment. In: IEEE Transactions on Software Engineering 21 (1995), Nr. 5, S. 563–575

8 Sandahl u. a. 1998 Sandahl, K. ; Blomkvist, O. ; Karlsson, J. ; Krysander, C. ; Lindvall, M. ; Ohlsson, N.: An extended replication of an experiment for assessing methods for software requirements inspections. In: Empirical Software Engineering 3 (1998), S. 327–354

9 Shull u. a. 2000 Shull, Forrest ; Rus, Ioana ; Basili, Victor: How Perspective-Based Reading Can Improve Requirements Inspections. In: IEEE Computer 33 (2000), Nr. 7, S. 73–79